Why you need queue sharing groups

gwydiontudur
Published on 30/07/2018 / Updated on 30/07/2018

High availability is very important for many of the customers that I talk to, and there are a number of features in MQ that help you achieve that goal. For example, replicated data queue managers on distributed platforms in IBM MQ V9.1 is getting a lot of interest. However, the platform with the highest level of availability is z/OS, and MQ is able to take advantage of the unique facilities in z/OS and the IBM Z hardware to provide a very highly available solution.

What is a queue sharing group?
If you’re using z/OS and concerned about high availability, you may already have a z/OS parallel sysplex. A parallel sysplex allows a group of z/OS systems to work together to provide a single system image that is highly available.
A queue sharing group (QSG), is a feature that’s unique to IBM MQ on z/OS, that takes advantage of the features provided by a z/OS parallel sysplex. It’s a group of queue managers that work together to provide a highly available environment.
With a queue manager in the QSG running on each z/OS system in the sysplex, you can shut down a queue manager, or an LPAR, without ever needing to take an outage of the entire QSG.

Shared queues
Of course, highly available applications don’t just need access to a queue manager at all times, they also need access to their queues. Shared queues provide the solution to that.
A shared queue is a queue that can be accessed by any member of the QSG. Both the queue itself, and the messages on the queue, can be accessed by any member of the QSG. This is possible as shared queues are stored in the coupling facility, where they can be accessed from any system in the sysplex.
As long as any queue manager in the QSG is available, the shared queue, and the messages on it, are available, and can be accessed by applications that connect to any member of the QSG.

There's also some clever processing that takes place should a queue manager disconnect abnormally from one of the coupling facility structures used by the QSG. The remaining queue managers in the QSG are able to complete work that was in progress on the queue manager that disconnected, to ensure consistency and availability of the messages on shared queues.

**An example**

To illustrate the benefits of a QSG, let’s take a look at the simple example below, where a client application on Linux uses MQ to make a request to a server application on z/OS, and receive a reply. In this example, the client application runs as an MQ client, so there’s no queue manager on Linux.
There are clearly several single points of failure here, which we can address with a QSG. The diagram below shows one way in which this can be done with a small QSG of two queue managers. Adding more queue managers to the QSG would increase both the availability and capacity of the system.

With shared queues, the client application can now access the same request and reply queues from any queue manager in the QSG that it is connected to. So how does it know which queue manager to connect to? The answer is that it doesn’t need to, as it now connects to the QSG, rather than a specific queue manager.

The ability to connect to any member of the QSG, rather than a specific queue manager, is provided by MQ shared channels, and technologies such as VIPA and Sysplex Distributor in z/OS which can distribute new connections into the QSG between the queue managers to balance the workload. The end result is that there is now no affinity between the application and a specific queue manager. Applications can connect to any queue manager in the QSG,
without knowing which queue manager they are connected to, and reconnect to another
queue manager in the QSG if that queue manager is shut down.
This example also illustrates some of the other benefits of a QSG. Messages arriving on the
request queue can be processed by any server application connected to a queue manager that
is a member of the QSG. This not only gives high availability, it also helps to spread the
workload between different systems, and makes it easier to add extra capacity. If you need to
add more capacity to the system, you can add more instances of the server application to
serve the same shared queues, and if necessary, add more queue managers to the QSG.

**Where to find more information?**
This article is only a basic introduction to QSGs. If you would like to learn more, take a look
at this topic in the MQ Knowledge Center.
**TAGS** HIGH AVAILABILITY, QSG
*by gwydiontudur*
3 comments on "Why you need queue sharing groups"

1. **MQnewtozos** August 13, 2018
   Do the messages reside in CF memory or on disk, or both? Please explain in simple
terms how to set up a queue in the CF – what a CFSTRUCT is and so on.
   Reply (Edit)
     - **gwydiontudur** August 15, 2018
       I’ve written another blog post which should answer your question. Take a look
   **jones** July 31, 2018
   ERROR org.springframework.jms.listener.DefaultMessageListenerContainer – Could
not refresh JMS Connection for destination 'queue:///QUEUE’ – retrying using
FixedBackOff{interval=5000, currentAttempts=28, maxAttempts=unlimited}. Cause:
MQJMS2005: failed to create MQQueueManager for 'ipaddress:queuemanager’;
nested exception is com.ibm.mq.MQException: MQJE001: An MQException
occurred: Completion Code 2, Reason 2009
Getting exception when trying to connection with mq